System Modelling and Design

Traffic Lights

Revision: 2, March 23, 2010

Ken Robinson

School of Computer Science & Engineering
The University of New South Wales, Sydney Australia

March 19, 2012

© Ken Robinson 2005-2010

mailto:k.robinson@unsw.edu.au
Outline

Objectives of this lecture

A simple 2-way intersection

The Context Machine

The Invariant
  A Safe state
  Strengthening the Invariant
  Other Invariants
  Some Theorems

The ChangeLight Event
  Sequencing
Part I

A simple 2-way intersection
Objectives of this lecture

- To explore the use of a state invariant to ensure safety.
- To explore the specification of some simple traffic light controllers for a simple traffic light controlled intersection.
- To expand safety to a general intersection.
- To model a traffic light controller for a general intersection.
Objectives of this lecture

- To explore the use of a state invariant to ensure safety.
- To explore the specification of some simple traffic light controllers for a simple traffic light controlled intersection.
- To expand safety to a general intersection.
- To model a traffic light controller for a general intersection.
Objectives of this lecture

- To explore the use of a state invariant to ensure safety.
- To explore the specification of some simple traffic light controllers for a simple traffic light controlled intersection.
- To expand safety to a general intersection.
- To model a traffic light controller for a general intersection.
Objectives of this lecture

- To explore the use of a state invariant to ensure safety.
- To explore the specification of some simple traffic light controllers for a simple traffic light controlled intersection.
- To expand safety to a general intersection.
- To model a traffic light controller for a general intersection.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing *Red*, *Green* and *Amber*, placed at *North*, *East*, *South* and *West* positions. The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

*Red* → *Green* → *Amber* → *Red* → …

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at North, East, South and West positions.

The North and South lights are always identical, as are the East and West lights.

There are no right-turn lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North, East, South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running North-South and the other East-West.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at North, East, South and West positions.

The North and South lights are always identical, as are the East and West lights.

There are no right-turn lights.

Lights should change in the sequence:
Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ... 

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing *Red*, *Green* and *Amber*, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

*Red* → *Green* → *Amber* → *Red* → ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red → Green → Amber → Red → ... 

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ …

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no right-turn lights.

Lights should change in the sequence:

Red $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ... 

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running North-South and the other East-West.

There are four sets of lights, each capable of showing Red, Green and Amber, placed at North, East, South and West positions.

The North and South lights are always identical, as are the East and West lights.

There are no right-turn lights.

Lights should change in the sequence:
Red → Green → Amber → Red → . . .

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
We will introduce a context machine containing the enumerated sets \textit{DIRECTION} and \textit{LIGHT}.

We will also specify a constant (function) \textit{OTHERDIR} that maps each direction to the other direction.

Context machines must be used in Event B to define sets and constants.
The Context Machine

We will introduce a context machine containing the enumerated sets \textit{DIRECTION} and \textit{LIGHT}.

We will also specify a constant (function) \textit{OTHERDIR} that maps each direction to the other direction.

Context machines must be used in Event B to define sets and constants.
The Context Machine

We will introduce a context machine containing the enumerated sets \textit{DIRECTION} and \textit{LIGHT}.

We will also specify a constant (function) \textit{OTHERDIR} that maps each direction to the other direction.

\textbf{Context machines must be used in Event B to define sets and constants.}
TwoWay_ctx.mch I

CONTEXT TrafficLights_ctx

SETS LIGHTS DIRECTION

CONSTANTS Red Green Amber NorthSouth EastWest OTHERDIR
AXIOMS

axm1: \( \text{LIGHTS} = \{ \text{Red, Green, Amber} \} \)

axm2: \( \text{Red} \neq \text{Green} \)

axm3: \( \text{Red} \neq \text{Amber} \)

axm4: \( \text{Green} \neq \text{Amber} \)

axm5: \( \text{DIRECTION} = \{ \text{NorthSouth, EastWest} \} \)

axm6: \( \text{NorthSouth} \neq \text{EastWest} \)

axm7: \( \text{OTHERDIR} \in \text{DIRECTION} \rightarrow \text{DIRECTION} \)

axm8: \( \text{OTHERDIR(NorthSouth)} = \text{EastWest} \)

axm9: \( \text{OTHERDIR(EastWest)} = \text{NorthSouth} \)
THEOREMS

\[ \text{thm1: } \forall d \cdot d \in \text{DIRECTION} \]
\[ \Rightarrow \]
\[ \text{OTHERDIR}(\text{OTHERDIR}(d)) = d \]

END
The above can be expressed more simply as

AXIOMS

axiom: \textit{partition}(\textit{LIGHTS}, \{\textit{Red}\}, \{\textit{Green}\}, \{\textit{Amber}\})

axiom: \textit{partition}(\textit{DIRECTION}, \{\textit{NorthSouth}\}, \{\textit{EastWest}\})
The Simple TwoWay Machine

The **TwoWay** machine

1. sees the context machine and has a state of one variable, *lights*, which is a total function from *DIRECTION* to *LIGHT*;
2. has a single event ChangeLight to change the lights;
3. ensures that lights change in the sequence Red, Green, Amber, ...;
4. maintains a safe intersection;
The Simple TwoWay Machine

The **TwoWay** machine

1. sees the context machine and has a state of one variable, *lights*, which is a total function from *DIRECTION* to *LIGHT*;
2. has a single event ChangeLight to change the lights;
3. ensures that lights change in the sequence Red, Green, Amber, ...;
4. maintains a safe intersection;
The Simple TwoWay Machine

The TwoWay machine

1. sees the context machine and has a state of one variable, *lights*, which is a total function from *DIRECTION* to *LIGHT*;

2. has a single event ChangeLight to change the lights;

3. ensures that lights change in the sequence Red, Green, Amber, …;

4. maintains a safe intersection;
The Simple TwoWay Machine

The **TwoWay** machine

1. sees the context machine and has a state of one variable, *lights*, which is a total function from *DIRECTION* to *LIGHT*;
2. has a single event ChangeLight to change the lights;
3. ensures that lights change in the sequence *Red, Green, Amber, …*;
4. maintains a safe intersection;
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be \textit{false}.

\[ \neg(safe) \implies \neg(invariant) \]  \hspace{1cm} (1)

3. Conversely, whenever the invariant is \textit{true}, the state should be safe.

\[ invariant \implies safe \] \hspace{1cm} (2)

4. Of course, 1 and 2 are equivalent; one is the \textit{contrapositive} of the other: \[ P \implies Q \equiv \neg Q \implies \neg P \]

5. If we have the \textit{weakest} invariant, then whenever the state is safe, the invariant will be \textit{true}.

6. We need to find adequately strong guards.
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be false.

\[ \neg(safe) \implies \neg(invariant) \] (1)

3. Conversely, whenever the invariant is true, the state should be safe.

\[ invariant \implies safe \] (2)

4. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \[ P \implies Q \equiv \neg Q \implies \neg P \]

5. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

6. We need to find adequately strong guards.
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be false.

\[
\neg (\text{safe}) \implies \neg (\text{invariant})
\]  

(1)

3. Conversely, whenever the invariant is true, the state should be safe.

\[
\text{invariant} \implies \text{safe}
\]

(2)

4. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \implies Q \equiv \neg Q \implies \neg P \)

5. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

6. We need to find adequately strong guards.
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be \textit{false}.
   \[
   \neg(\text{safe}) \implies \neg(\text{invariant}) \tag{1}
   \]
3. Conversely, whenever the invariant is \textit{true}, the state should be safe.
   \[
   \text{invariant} \implies \text{safe} \tag{2}
   \]
4. Of course, 1 and 2 are equivalent; one is the \textit{contrapositive} of the other: \( P \implies Q \equiv \neg Q \implies \neg P \)
5. If we have the \textit{weakest} invariant, then whenever the state is safe, the invariant will be \textit{true}.
6. We need to find adequately strong guards.
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be false.
   \[ \neg (\text{safe}) \implies \neg (\text{invariant}) \] (1)
3. Conversely, whenever the invariant is true, the state should be safe.
   \[ \text{invariant} \implies \text{safe} \] (2)
4. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \implies Q \equiv \neg Q \implies \neg P \)
5. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.
6. We need to find adequately strong guards.
The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be \textit{false}.

\[ \neg(\text{safe}) \implies \neg(\text{invariant}) \] (1)

3. Conversely, whenever the invariant is \textit{true}, the state should be safe.

\[ \text{invariant} \implies \text{safe} \] (2)

4. Of course, 1 and 2 are equivalent; one is the \textit{contrapositive} of the other: \( P \implies Q \equiv \neg Q \implies \neg P \)

5. If we have the \textit{weakest} invariant, then whenever the state is safe, the invariant will be \textit{true}.

6. We need to find adequately strong guards.
1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be false.
   \[ \neg(safe) \implies \neg(invariant) \]  
   \[ (1) \]
3. Conversely, whenever the invariant is true, the state should be safe.
   \[ invariant \implies safe \]  
   \[ (2) \]
4. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \implies Q \equiv \neg Q \implies \neg P \)
5. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.
6. We need to find adequately strong guards.
A Safe state

The state invariant should be:

1. false for all unsafe states   true for all safe states
2. An initial attempt at an invariant might be

$$\neg(lights(NorthSouth) = \text{Green} \land lights(EastWest) = \text{Green})$$

which, using the predicate identities

$$\neg(P \land Q) = \neg P \lor \neg Q = P \implies \neg Q$$

may be written

$$lights(NorthSouth) = \text{Green} \implies \neg(lights(EastWest) = \text{Green})$$

If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. *false* for all unsafe states    *true* for all safe states

2. An initial attempt at an invariant might be

   \[-(\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green})\]

   which, using the predicate identities

   \[-(P \land Q) = \neg P \lor \neg Q = P \implies \neg Q.\]

   may be written

   \[\text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg(\text{lights}(\text{EastWest}) = \text{Green})\]

   If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. **false** for all unsafe states  **true** for all safe states
2. An initial attempt at an invariant might be

   \[ \neg (\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green}) \]

   which, using the predicate identities

   \[ \neg (P \land Q) = \neg P \lor \neg Q = P \Rightarrow \neg Q \]

   may be written

   \[ \text{lights}(\text{NorthSouth}) = \text{Green} \Rightarrow \neg (\text{lights}(\text{EastWest}) = \text{Green}) \]

   If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. \textit{false} for all unsafe states \hspace{1em} \textit{true} for all safe states
2. An initial attempt at an invariant might be

\[ \neg (\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green}) \]

which, using the predicate identities

\[ \neg (P \land Q) = \neg P \lor \neg Q = P \implies \neg Q , \]

may be written

\[ \text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg (\text{lights}(\text{EastWest}) = \text{Green}) \]

If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. *false* for all unsafe states   *true* for all safe states
2. An initial attempt at an invariant might be

\[ \neg (\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green}) \]

which, using the predicate identities

\[ \neg (P \land Q) \equiv \neg P \lor \neg Q \equiv P \implies \neg Q, \]

may be written

\[ \text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg (\text{lights}(\text{EastWest}) = \text{Green}) \]

If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. *false* for all unsafe states  *true* for all safe states
2. An initial attempt at an invariant might be

\[ \neg (lights(NorthSouth) = \text{Green} \land lights(EastWest) = \text{Green}) \]

which, using the predicate identities

\[ \neg(P \land Q) \equiv \neg P \lor \neg Q \equiv P \implies \neg Q , \]

may be written

\[ lights(NorthSouth) = \text{Green} \implies \neg(lights(EastWest) = \text{Green}) \]

If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. *false* for all unsafe states  
   *true* for all safe states

2. An initial attempt at an invariant might be

   $\neg (\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green})$

   which, using the predicate identities

   $\neg (P \land Q) \equiv \neg P \lor \neg Q \equiv P \implies \neg Q$

   may be written

   $\text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg (\text{lights}(\text{EastWest}) = \text{Green})$

If used as a term in a larger predicate, the above implication may need to be parenthesised.
A Safe state

The state invariant should be:

1. \textit{false} for all unsafe states \quad \textit{true} for all safe states
2. An initial attempt at an invariant might be

\[ \neg (\text{lights}(\text{NorthSouth}) = \text{Green} \land \text{lights}(\text{EastWest}) = \text{Green}) \]

which, using the predicate identities

\[ \neg(P \land Q) \equiv \neg P \lor \neg Q \equiv P \implies \neg Q , \]

may be written

\[ \text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg(\text{lights}(\text{EastWest}) = \text{Green}) \]

If used as a term in a larger predicate, the above implication may need to be parenthesised.
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[ \neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \]

\[ \equiv \]

\[ (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})) \]

\[ \equiv \]

\[ (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{EastWest}) = \text{Red}) \] (3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

\[
\begin{array}{ccc}
\text{Red} & \text{Green} & \text{Amber} \\
\text{Red} & \text{safe} & \text{safe} & \text{safe} \\
\text{Green} & \text{safe} & \text{unsafe} & \text{unsafe} \\
\text{Amber} & \text{safe} & \text{unsafe} & \text{unsafe} \\
\end{array}
\]

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\}) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\})) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \\
\text{lights}(\text{EastWest}) = \text{Red}) 
\]

(3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

$$\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})$$

$$\equiv$$

$$\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \implies \neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\})$$

$$\equiv$$

$$\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{EastWest}) = \text{Red}$$

(3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

$$
\neg (\text{lights} (\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \\
\text{lights} (\text{EastWest}) \in \{\text{Green, Amber}\})
\equiv

\neg (\text{lights} (\text{EastWest}) \in \{\text{Green, Amber}\})

\equiv

\text{lights} (\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow

\text{lights} (\text{EastWest}) = \text{Red}
$$

(3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg \left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\} \right) \\
\equiv \\
\left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \neg \left( \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\} \right) \right) \\
\equiv \\
\left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \text{lights}(\text{EastWest}) = \text{Red} \right) \tag{3}
\]
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[ \neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})) \equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{EastWest}) = \text{Red}) \] (3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \\
\text{lights}(\text{EastWest}) = \text{Red})
\] (3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green nor Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[ \neg ( \text{lights} (\text{NorthSouth}) \in \{ \text{Green, Amber} \} \land \text{lights} (\text{EastWest}) \in \{ \text{Green, Amber} \} ) \]

\[ \equiv \]

\[ ( \text{lights} (\text{NorthSouth}) \in \{ \text{Green, Amber} \} \Rightarrow \neg ( \text{lights} (\text{EastWest}) \in \{ \text{Green, Amber} \} ) ) \]

\[ \equiv \]

\[ ( \text{lights} (\text{NorthSouth}) \in \{ \text{Green, Amber} \} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red} ) \]  (3)
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg \left( \text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\} \right)
\]

\[
\equiv
\]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \neg(\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}))
\]

\[
\equiv
\]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{EastWest}) = \text{Red})
\]
Strengthening the Invariant

Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})
\]

\[\equiv \]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}))
\]

\[\equiv \]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \implies \\
\text{lights}(\text{EastWest}) = \text{Red})
\]

(3)
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[
\text{lights} (\text{NorthSouth}) = \text{Red} \lor \text{lights} (\text{EastWest}) = \text{Red}
\]

\[
\text{Red} \in \text{ran} (\text{lights})
\]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[
\forall \text{dir.} \; \text{dir} \in \text{DIRECTION} \land \text{lights}(\text{dir}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{OTHERDIR} (\text{dir})) = \text{Red}
\]
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[
lights(\text{NorthSouth}) = \text{Red} \lor lights(\text{EastWest}) = \text{Red}
\]

\[
\text{Red} \in \text{ran}(lights)
\]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[
\forall \text{dir}. \text{dir} \in \text{DIRECTION} \land lights(\text{dir}) \in \{\text{Green}, \text{Amber}\} \implies lights(\text{OTHERDIR}(\text{dir})) = \text{Red}
\]
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[
\text{\textit{lights}}(\text{\textit{NorthSouth}}) = \text{\textit{Red}} \lor \text{\textit{lights}}(\text{\textit{EastWest}}) = \text{\textit{Red}}
\]

\[
\text{\textit{Red}} \in \text{ran}(\text{\textit{lights}})
\]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[
\forall \text{dir}. \text{dir} \in \text{DIRECTION} \land \text{\textit{lights}}(\text{dir}) \in \{\text{Green, Amber}\} \implies \text{\textit{lights}}(\text{\textit{OTHERDIR}}(\text{dir})) = \text{\textit{Red}}
\]
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[
\text{lights}(\text{NorthSouth}) = \text{Red} \lor \text{lights}(\text{EastWest}) = \text{Red}
\]

\[
\text{Red} \in \text{ran(lights)}
\]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[
\forall \text{dir}. \, \text{dir} \in \text{DIRECTION} \land \text{lights}(\text{dir}) \in \{\text{Green, Amber}\} \implies \\
\text{lights}(\text{OTHERDIR(dir)}) = \text{Red}
\]
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[
lights(\text{NorthSouth}) = \text{Red} \lor \lights(\text{EastWest}) = \text{Red}
\]

\[
\text{Red} \in \text{ran}(\lights)
\]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[
\forall \text{dir}. \text{dir} \in \text{DIRECTION} \land \lights(\text{dir}) \in \{\text{Green, Amber}\} \implies \lights(\text{OTHERDIR}(\text{dir})) = \text{Red}
\]

(4)
Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

\[ \text{lights}(\text{NorthSouth}) = \text{Red} \lor \text{lights}(\text{EastWest}) = \text{Red} \]

\[ \text{Red} \in \text{ran}(\text{lights}) \]

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

\[ \forall \text{dir}. \text{dir} \in \text{DIRECTION} \land \text{lights}(\text{dir}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{OTHERDIR}(\text{dir})) = \text{Red} \] (4)
Some Theorems

In this model of **TwoWay** we add the following alternative formulations of the invariant as theorems:

- \( \forall \text{dir}. (\text{dir} \in DIRECTION \land \text{lights}(\text{dir}) \in \{\text{Green, Amber}\}) \implies \text{lights}(\text{OTHERDIR}(\text{dir})) = \text{Red} \)

- \( \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\} \implies \text{lights}(\text{NorthSouth}) = \text{Red} \)

Theorems contain properties that are implied by the invariant, and hence discharging the proof obligations for the assertions proves that each conjunct in the assertions is implied by the invariant.
The ChangeLight Event

The ChangeLight event will have two parameters \( \text{dir} \) and \( \text{light} \). The event will change the lights in direction \( \text{dir} \) to colour \( \text{light} \)

\[
lights(\text{dir}) := \text{light}
\]

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

\[
(\text{light} \in \{\text{Green}, \text{Amber}\} \implies \lights(\text{OTHERDIR}(\text{dir})) = \text{Red})
\]
The ChangeLight Event

The ChangeLight event will have two parameters \( \text{dir} \) and \( \text{light} \).

The event will change the lights in direction \( \text{dir} \) to colour \( \text{light} \)

\[
lights(\text{dir}) := \text{light}
\]

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

\[
(\text{light} \in \{\text{Green, Amber}\} \implies \\
\quad \lights(\text{OTHERDIR}(\text{dir})) = \text{Red})
\]
The ChangeLight Event

The ChangeLight event will have two parameters $dir$ and $light$. The event will change the lights in direction $dir$ to colour $light$

$$\text{lights}(dir) := light$$

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

$$(\text{light} \in \{\text{Green}, \text{Amber}\} \implies \text{lights}(\text{OTHERDIR}(dir)) = \text{Red})$$
The ChangeLight Event

The ChangeLight event will have two parameters \( dir \) and \( light \). The event will change the lights in direction \( dir \) to colour \( light \)

\[
lights(dir) := light
\]

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

\[
(light \in \{\text{Green, Amber}\} \implies \lights(\text{OTHERDIR}(dir)) = \text{Red})
\]
The ChangeLight Event

The ChangeLight event will have two parameters $dir$ and $light$. The event will change the lights in direction $dir$ to colour $light$

$$\text{lights}(dir) := light$$

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

$$(light \in \{\text{Green, Amber}\} \implies \text{lights}(\text{OTHERDIR}(dir)) = \text{Red})$$
The ChangeLight Event

The ChangeLight event will have two parameters \( \text{dir} \) and \( \text{light} \). The event will change the lights in direction \( \text{dir} \) to colour \( \text{light} \)

\[
lights(\text{dir}) := \text{light}
\]

The guards need to be strong enough to ensure both safety and correct sequencing. Safety can be ensured with the guard:

\[
(\text{light} \in \{ \text{Green, Amber} \} \implies \lights(\text{OTHERDIR}(\text{dir})) = \text{Red})
\]
Sequencing

Changing to

Red: the current colour should be Amber, so
\[ \text{light} = \text{Red} \implies \text{lights(dir)} = \text{Amber} \]

Green: the current colour should be Red, so
\[ \text{light} = \text{Green} \implies \text{lights(dir)} = \text{Red} \]

Amber: the current colour should be Green, so
\[ \text{light} = \text{Amber} \implies \text{lights(dir)} = \text{Green} \]

Notice that the sequencing guards for both Red and Amber together with the invariant ensure safety, we only need to be concerned with safety for changing to Green.
Sequencing

Changing to

Red: the current colour should be Amber, so
\[ \text{light} = \text{Red} \implies \text{lights(dir)} = \text{Amber} \]

Green: the current colour should be Red, so
\[ \text{light} = \text{Green} \implies \text{lights(dir)} = \text{Red} \]

Amber: the current colour should be Green, so
\[ \text{light} = \text{Amber} \implies \text{lights(dir)} = \text{Green} \]

Notice that the sequencing guards for both Red and Amber together with the invariant ensure safety, we only need to be concerned with safety for changing to Green.
Sequencing

Changing to

**Red**: the current colour should be *Amber*, so
\[ \text{light} = \text{Red} \implies \text{lights(dir)} = \text{Amber} \]

**Green**: the current colour should be *Red*, so
\[ \text{light} = \text{Green} \implies \text{lights(dir)} = \text{Red} \]

**Amber**: the current colour should be *Green*, so
\[ \text{light} = \text{Amber} \implies \text{lights(dir)} = \text{Green} \]

Notice that the sequencing guards for both *Red* and *Amber* together with the invariant ensure safety, we only need to be concerned with safety for changing to *Green*. 
Sequencing

Changing to

**Red:** the current colour should be **Amber**, so
\[ light = \text{Red} \implies \text{lights}(\text{dir}) = \text{Amber} \]

**Green:** the current colour should be **Red**, so
\[ light = \text{Green} \implies \text{lights}(\text{dir}) = \text{Red} \]

**Amber:** the current colour should be **Green**, so
\[ light = \text{Amber} \implies \text{lights}(\text{dir}) = \text{Green} \]

Notice that the sequencing guards for both **Red** and **Amber** together with the invariant ensure safety, we only need to be concerned with safety for changing to **Green**.
Sequencing

Changing to

**Red:** the current colour should be *Amber*, so
\[ light = \text{Red} \implies \text{lights}(\text{dir}) = \text{Amber} \]

**Green:** the current colour should be *Red*, so
\[ light = \text{Green} \implies \text{lights}(\text{dir}) = \text{Red} \]

**Amber:** the current colour should be *Green*, so
\[ light = \text{Amber} \implies \text{lights}(\text{dir}) = \text{Green} \]

Notice that the sequencing guards for both *Red* and *Amber* together with the invariant ensure safety, we only need to be concerned with safety for changing to *Green*. 
The final guards that ensure both safety and correct sequencing are:

- \( \text{light} = \text{Red} \implies \text{lights(dir)} = \text{Amber} \)
- \( \text{light} = \text{Green} \implies \text{lights(dir)} = \text{Red} \)
- \( \text{light} = \text{Green} \implies \text{lights(\text{OTHERDIR(dir)})} = \text{Red} \)
- \( \text{light} = \text{Amber} \implies \text{lights(dir)} = \text{Green} \)
Final Guards for ChangeLight

The final guards that ensure both safety and correct sequencing are:

\[
\begin{align*}
\text{light} = \text{Red} & \implies \text{lights}(\text{dir}) = \text{Amber} \\
\text{light} = \text{Green} & \implies \text{lights}(\text{dir}) = \text{Red} \\
\text{light} = \text{Green} & \implies \text{lights}(\text{OTHERDIR}(\text{dir})) = \text{Red} \\
\text{light} = \text{Amber} & \implies \text{lights}(\text{dir}) = \text{Green}
\end{align*}
\]
Final Guards for ChangeLight

The final guards that ensure both safety and correct sequencing are:

- \( \text{light} = \text{Red} \quad \implies \quad \text{lights}(\text{dir}) = \text{Amber} \)
- \( \text{light} = \text{Green} \quad \implies \quad \text{lights}(\text{dir}) = \text{Red} \)
- \( \text{light} = \text{Green} \quad \implies \quad \text{lights}(\text{OTHERDIR}(\text{dir})) = \text{Red} \)
- \( \text{light} = \text{Amber} \quad \implies \quad \text{lights}(\text{dir}) = \text{Green} \)
Final Guards for ChangeLight

The final guards that ensure both safety and correct sequencing are:

\[
\begin{align*}
\text{light} = \text{Red} & \implies \text{lights(dir)} = \text{Amber} \\
\text{light} = \text{Green} & \implies \text{lights(dir)} = \text{Red} \\
\text{light} = \text{Green} & \implies \text{lights(OTHERDIR(dir))} = \text{Red} \\
\text{light} = \text{Amber} & \implies \text{lights(dir)} = \text{Green}
\end{align*}
\]
Final Guards for ChangeLight

The final guards that ensure both safety and correct sequencing are:

\[
\begin{align*}
light &= \text{Red} & \implies & \text{lights(dir)} &= \text{Amber} \\
light &= \text{Green} & \implies & \text{lights(dir)} &= \text{Red} \\
light &= \text{Green} & \implies & \text{lights(OTHERDIR(dir))} &= \text{Red} \\
light &= \text{Amber} & \implies & \text{lights(dir)} &= \text{Green}
\end{align*}
\]
The ChangeLight Event I

MACHINE ChangeLights
SEES TrafficLights_ctx
VARIABLES
\[ lights \]

INVAR\[\text{I}NTANTS\]
\[ \text{inv1: } lights \in \text{DIRECTION} \rightarrow \text{LIGHTS} \]
\[ \text{inv2: } lights(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow lights(\text{EastWest}) = \text{Red} \]
EVENTS

Initialisation

begin

act1: \textit{lights} := \{\textit{NorthSouth} \mapsto \text{Red}, \textit{EastWest} \mapsto \text{Red}\}

end
Event \( \text{ToAmber} \triangleq \)

any \( \text{dir} \)

when

\( \text{grd1}: \) \( \text{dir} \in \text{DIRECTION} \)

\( \text{grd2}: \) \( \text{lights}(\text{dir}) = \text{Green} \)

then

\( \text{act1}: \) \( \text{lights}(\text{dir}):= \text{Amber} \)

end
The ChangeLight Event IV

Event  \textit{ToGreen} \triangleq \\
\text{any} \quad \textit{dir} \\
\text{when} \\
\textit{grd1}: \quad \textit{dir} \in \textit{DIRECTION} \\
\textit{grd2}: \quad \text{lights}(\text{OTHERDIR}(\textit{dir})) = \text{Red} \\
\textit{grd3}: \quad \text{lights}(\textit{dir}) = \text{Red} \\
\text{then} \\
\textit{act1}: \quad \text{lights}(\textit{dir}) := \text{Green} \\
\text{end}
The ChangeLight Event V

Event  \( \text{ToRed} \doteq \)

any  \( \text{dir} \)

when

\( \text{grd1:} \quad \text{dir} \in \text{DIRECTION} \)

\( \text{grd2:} \quad \text{lights(dir)} = \text{Amber} \)

then

\( \text{act1:} \quad \text{lights(dir)} := \text{Red} \)

end

END
Part II

A general multi-way intersection
A General Multi-Way Intersection

The New Safety Invariant

A Traffic Light Controller
A General Multi-Way Intersection

We will now expand to a completely general intersection with many directions, denoted by the finite set $DIRECTION$. To define directions that conflict with one another we will use a relation $CONFLICT$ that maps each direction to all other directions, with which it conflicts. Clearly, $CONFLICT$ must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if $dir_1$ conflicts with $dir_2$ then $dir_2$ conflicts with $dir_1$. 
We will now expand to a completely general intersection with many directions, denoted by the finite set $DIRECTION$. To define directions that conflict with one another we will use a relation $CONFLICT$ that maps each direction to all other directions, with which it conflicts. Clearly, $CONFLICT$ must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if $dir_1$ conflicts with $dir_2$ then $dir_2$ conflicts with $dir_1$. 
We will now expand to a completely general intersection with many directions, denoted by the finite set $DIRECTION$. To define directions that conflict with one another we will use a relation $CONFLICT$ that maps each direction to all other directions, with which it conflicts. Clearly, $CONFLICT$ must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if $dir_1$ conflicts with $dir_2$ then $dir_2$ conflicts with $dir_1$. 
We will now expand to a completely general intersection with many directions, denoted by the finite set \( DIRECTION \). To define directions that conflict with one another we will use a relation \( CONFLICT \) that maps each direction to all other directions, with which it conflicts. Clearly, \( CONFLICT \) must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if \( dir_1 \) conflicts with \( dir_2 \) then \( dir_2 \) conflicts with \( dir_1 \).
We will now expand to a completely general intersection with many directions, denoted by the finite set $DIRECTION$. To define directions that conflict with one another we will use a relation $CONFLICT$ that maps each direction to all other directions, with which it conflicts. Clearly, $CONFLICT$ must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if $dir_1$ conflicts with $dir_2$ then $dir_2$ conflicts with $dir_1$. 
The New Safety Invariant

The safety invariant for our generalised intersection is

$$\forall d. d \in DIRECTION \land lights(d) \in \{\text{Green, Amber}\} \implies $$

$$lights[\text{CONFLICT}[\{d\}] = \{\text{Red}\}$$

(5)

This can be seen as a generalisation of the earlier safety invariant (4).
TrafficLights1_ctx

CONTEXT  TrafficLights1_ctx

SETS

LIGHTS
DIRECTION
A General Multi-Way Intersection

TrafficLights1_ctx II

CONSTANTS

Red
Green
Amber
CONFLICT
adir
TrafficLights1_ctx III

AXIOMS

\textit{axm1}: \textit{partition(LIGHTS, \{Red\}, \{Green\}, \{Amber\})} \\
\text{Three light colours}

\textit{axm2}: \textit{finite(DIRECTION)} \text{ \textit{DIRECTION is a finite set of directions}}

\textit{axm3}: \textit{CONFLICT} \in \textit{DIRECTION} \leftrightarrow \textit{DIRECTION} \text{ \textit{CONFLICT relates conflicting directions}}

\textit{axm4}: \textit{CONFLICT} \cap (\textit{DIRECTION} \triangleleft \textit{id}) = \emptyset \text{ \textit{a direction cannot conflict with itself}}

\textit{axm5}: \textit{CONFLICT}^{-1} = \textit{CONFLICT} \text{ \textit{conflicts are symmetric}}

\textit{thm1}: \forall d \cdot d \in \textit{DIRECTION} \Rightarrow d \notin \textit{CONFLICT}[[\{d\}]]

\textit{thm2}: \forall d1, d2 \cdot d1 \notin \textit{CONFLICT}[[\{d2\}] \Rightarrow d2 \notin \textit{CONFLICT}[[\{d1\}]]
TrafficLights1_ctx IV

axm6: \[ \text{adir} \in \text{DIRECTION} \]

any direction (used as a parameter)

END
We will now model a traffic light controller that responds to the following requirements:

1. A controller is required that can be used to control the whole of a general intersection.
2. The controller must be able to set the light in any direction to Green and must do so in a safe transition sequence.
3. The intersection must always be in a safe state.
We will now model a traffic light controller that responds to the following requirements:

1. A controller is required that can be used to control the whole of a general intersection.

2. The controller must be able to set the light in any direction to Green and must do so in a safe transition sequence.

3. The intersection must always be in a safe state.
We will now model a traffic light controller that responds to the following requirements:

1. A controller is required that can be used to control the whole of a general intersection.
2. The controller must be able to set the light in any direction to Green and must do so in a safe transition sequence.
3. The intersection must always be in a safe state.
A Traffic Light Controller

We will now model a traffic light controller that responds to the following requirements:

1. A controller is required that can be used to control the whole of a general intersection.
2. The controller must be able to set the light in any direction to **Green** and must do so in a safe transition sequence.
3. The intersection must always be in a safe state.
Our Response

Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $adir$, to Green, setting all lights in conflicting directions to Red;
2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;
3. to refine that machine to model the process of changing the state while maintaining a safe state.
4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $adir$, to Green, setting all lights in conflicting directions to Red;

2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;

3. to refine that machine to model the process of changing the state while maintaining a safe state.

4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $adir$, to Green, setting all lights in conflicting directions to Red;

2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;

3. to refine that machine to model the process of changing the state while maintaining a safe state.

4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $\text{adir}$, to Green, setting all lights in conflicting directions to Red;

2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;

3. to refine that machine to model the process of changing the state while maintaining a safe state.

4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $adir$, to Green, setting all lights in conflicting directions to Red;

2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;

3. to refine that machine to model the process of changing the state while maintaining a safe state.

4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
Our Response

Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant $adir$, to Green, setting all lights in conflicting directions to Red;

2. to model the top-level state as having only Red and Green, these being the “steady state” light colours;

3. to refine that machine to model the process of changing the state while maintaining a safe state.

4. towards safety, the transition from Green to Red will introduce the light colour Amber and delays between light transitions.
MACHINE ChangeLight1
SEES TrafficLights1_ctx
VARIABLES
   lights
IN INVARIANTS
   inv1: lights ∈ DIRECTION → {Red, Green}
   inv2: ∀ d · d ∈ DIRECTION ∧ lights(d) = Green 
       ⇒ lights[CONFLICT[{d}]] ⊆ {Red}
EVENTS

Initialisation

begin

act1: \( lights : | \)
\( lights' \in DIRECTION \rightarrow \{ Red, Green \} \)
\( \land (\forall d \cdot d \in DIRECTION \land \) \)
\( \land lights'(d) = Green \)
\( \Rightarrow lights'[\text{CONFLICT}[\{d\}]] \subseteq \{ Red \} \)

end
Event \( ToGreen \) \( \triangleq \)

when

\( grd1: \) \( \text{lights}(adir) = Red \)

then

\( act1: \) \( \text{lights} \leftarrow \text{lights} \)

\[ \triangleleft (CONFLICT[\{adir\}] \times \{Red\}) \]

\[ \triangleleft \{adir \mapsto Green\} \]

end

END
The first stage is still abstract and would be very unsafe, if implemented literally according to the model, as it is not safe to change lights instantaneously from Green to Red. This refinement introduces Amber into the lights sequence and the following slave events:

- **ToAmber**: changes, in arbitrary sequence, all the currently Green lights in conflicting directions to Amber;
- **ToRed**: changes the Amber lights to Red;
- **Delay**: models a delay between Amber and Red and between Red and the final setting of the light in the adir direction to Green.

The refinement of ToGreen waits until all the conflicting directions have been changed to Red and then sets the light in the adir direction to Green.
The first stage is still abstract and would be very unsafe, if implemented literally according to the model, as it is not safe to change lights instantaneously from Green to Red. This refinement introduces Amber into the lights sequence and the following slave events:

- **ToAmber**: changes, in arbitrary sequence, all the currently Green lights in conflicting directions to Amber;
- **ToRed**: changes the Amber lights to Red;
- **Delay**: models a delay between Amber and Red and between Red and the final setting of the light in the adir direction to Green.

The refinement of ToGreen waits until all the conflicting directions have been changed to Red and then sets the light in the adir direction to Green.
The first stage is still abstract and would be very unsafe, if implemented literally according to the model, as it is not safe to change lights instantaneously from Green to Red. This refinement introduces Amber into the lights sequence and the following slave events:

- **ToAmber**: changes, in arbitrary sequence, all the currently Green lights in conflicting directions to Amber;
- **ToRed**: changes the Amber lights to Red;
- **Delay**: models a delay between Amber and Red and between Red and the final setting of the light in the adir direction to Green.

The refinement of ToGreen waits until all the conflicting directions have been changed to Red and then sets the light in the adir direction to Green.
MACHINE ChangeLight1R
REFINES ChangeLight1
SEES TrafficLights1_ctx
VARIABLES
xlights
Extended lights, Red, Green and Amber lights
delay
delay between Amber and Red, and between Red and Green
ChangeLight1R refinement II

INVARIANTS

\[ \text{inv1: } xlights \in \text{DIRECTION} \rightarrow \text{LIGHTS} \]

\[ \text{inv2: } \forall d \cdot d \in \text{DIRECTION} \land xlights[\{d\}] \subseteq \{\text{Green, Amber}\} \Rightarrow xlights[\text{CONFLICT}[\{d\}]] \subseteq \{\text{Red}\} \]

\[ \text{inv3: } xlights \Leftrightarrow (\text{CONFLICT}[\{\text{adir}\}] \times \{\text{Red}\}) = \text{lights} \Leftrightarrow (\text{CONFLICT}[\{\text{adir}\}] \times \{\text{Red}\}) \]

\[ \text{inv4: } xlights(\text{adir}) = \text{Green} \Rightarrow \text{lights} = xlights \]

\[ \text{inv5: } \text{delay} \subseteq \text{DIRECTION} \]
Changelight1r Refinement III

**Thm 1:** \( \text{finite}(xlights) \)

**Thm 2:**

\[
\text{CONFLICT}\left[\{\text{adir}\}\right] \ll lights = \text{CONFLICT}\left[\{\text{adir}\}\right] \ll xlights
\]

**Thm 3:**

\[
\forall d, b, a \cdot d \in \text{DIRECTION} \\
\wedge b \in \text{LIGHTS} \wedge a \in \text{LIGHTS} \wedge a \neq b \wedge xlights(d) = b \\
\Rightarrow \\
\text{card}\left(\left(xlights \ll \{d \mapsto a\}\right) \triangleright \{b\}\right) \\
= \\
\text{card}\left(xlights \triangleright \{b\}\right) - 1
\]

changing light in direction d from b (= before) to a (= after) decreases number of colour b lights by 1
ChangeLight1R refinement IV

thm4:
\[ \forall d, b, a \cdot d \in \text{DIRECTION} \land b \in \text{LIGHTS} \land a \in \text{LIGHTS} \land a \neq b \land xlights(d) = b \Rightarrow \]
\[ \text{card}( (\{d \mapsto a\} \uplus \{a\}) ) = \text{card}(xlights \uplus \{a\}) + 1 \]
changing light in direction d from b (= before) to a (=after) increases number of colour a lights by 1
thm5: \[ \forall d, b, a, c \cdot d \in \text{DIRECTION} \]
\[ \land b \in \text{LIGHTS} \land a \in \text{LIGHTS} \land c \in \text{LIGHTS} \]
\[ \land xlights(d) = b \land c \neq a \land c \neq b \]
⇒
\[ \text{card}((xlights \leftrightarrow \{ d \mapsto a \}) \triangleright \{ c \}) \]
=  
\[ \text{card}(xlights \triangleright \{ c \}) \]
changing light in direction \( d \) from \( b \) (= before) to \( a \) (=after) does not change number of colour \( c \), \( c \neq a \), \( c \neq b \)
EVENTS
Initialisation
begin
with

lights': \textit{lights'} = x\textit{lights'}

\textbf{act1:} \quad x\textit{lights} : |
\quad x\textit{lights}' \in DIRECTION \rightarrow \{\textit{Red}, \textit{Green}\}
\quad \land (\forall d \cdot d \in DIRECTION
\quad \land x\textit{lights}'(d) = \textit{Green}
\quad \Rightarrow
\quad x\textit{lights}'[\textit{CONFLICT}[[d]]] \subseteq \{\textit{Red}\})

\textbf{act2:} \quad delay := \emptyset

end
Event $\text{ToGreen} \equiv$
refines $\text{ToGreen}$

when

$\text{grd1: } xlights(adir) = \text{Red}$

$\text{grd2: } xlights[\text{CONFLICT}\{adir\}] \subseteq \{\text{Red}\}$

$\text{grd3: } delay = \emptyset$

then

$\text{act1: } xlights := xlights \leftrightarrow \{adir \mapsto \text{Green}\}$

end
Event $\text{GreenToAmber} \equiv$

Status convergent

any dir when

grd1: $\text{dir} \in \text{CONFLICT}[\{\text{adir}\}]$

grd2: $\text{xlights}(\text{dir}) = \text{Green}$

grd3: $\text{xlights}(\text{adir}) \neq \text{Green}$

grd4: $\text{adir} \notin \text{delay}$

then

act1: $\text{xlights}(\text{dir}) := \text{Amber}$

act2: $\text{delay} := \text{delay} \cup \{\text{dir}\}$

end
ChangeLight1R refinement IX

Event \( AmberToRed \subseteq \)
Status convergent

any \( dir \)

when

\( grd1: \) \( dir \in CONFLICT[\{adir\}] \)

\( grd2: \) \( xlights(dir) = Amber \)

\( grd3: \) \( dir \notin delay \)

\( grd4: \) \( xlights(adir) \neq Green \)

then

\( act1: \) \( xlights(dir) := Red \)

\( act2: \) \( delay := delay \cup \{dir\} \)

end
Event \( Delay \supseteq \)

Status convergent

\( \text{any } \text{dir} \)

when

\( grd1 : \text{dir} \in \text{delay} \)

then

\( act1 : \text{delay} := \text{delay} \setminus \{ \text{dir} \} \)

end
VARIANT

\[ 4 \times \text{card}(\text{xlights} \uparrow \{\text{Green}\}) \]
\[ + 2 \times \text{card}(\text{xlights} \uparrow \{\text{Amber}\}) \]
\[ + \text{card}(\text{delay}) \]

END
The events *GreenToAmber*, *AmberToRed* and *Delay* are *slave* events; that is they are not the main event, which is *ToGreen*. Consequently they must not be able to be enabled forever; that is they must be *convergent*.

In EventB this is verified by providing a *variant* expression: an expression that has a lower bound and must strictly decrease every time a convergent event fires.

In this model the variant is an expression that yields a natural number —hence lower bound 0— that strictly decreases.
The events *GreenToAmber*, *AmberToRed* and *Delay* are slave events; that is they are not the main event, which is *ToGreen*. Consequently they must not be able to be enabled forever; that is they must be convergent.

In EventB this is verified by providing a variant expression: an expression that has a lower bound and must strictly decrease everytime a convergent event fires.

In this model the variant is an expression that yields a natural number —hence lower bound 0— that strictly decreases.
The events *GreenToAmber*, *AmberToRed* and *Delay* are *slave* events; that is they are not the main event, which is *ToGreen*.

Consequently they must not be able to be enabled forever; that is they must be *convergent*.

In EventB this is verified by providing a *variant* expression: an expression that has a lower bound and must strictly decrease everytime a convergent event fires.

In this model the variant is an expression that yields a natural number —hence lower bound 0— that strictly decreases.
The events *GreenToAmber*, *AmberToRed* and *Delay* are *slave* events; that is they are not the main event, which is *ToGreen*.

Consequently they must not be able to be enabled forever; that is they must be *convergent*.

In EventB this is verified by providing a *variant* expression: an expression that has a lower bound and must strictly decrease everytime a convergent event fires.

In this model the variant is an expression that yields a natural number —hence lower bound 0— that strictly decreases.
The Variant

The variant is a sum of the following:

- 4 times the number of Red lights;
- 2 times the number of Amber lights;
- 1 times the number of current delays

This is derived from the GreenToAmber event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the AmberToRed event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- **4 times** the number of **Red** lights;
- **2 times** the number of **Amber** lights;
- **1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1;

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

4 times the number of Red lights;
2 times the number of Amber lights;
1 times the number of current delays

This is derived from the GreenToAmber event that

• decreases the number of Red lights by 1;
• increases the number of Amber lights by 1;
• increases the number of delays by 1.

and the AmberToRed event that

• decreases the number of Amber lights by 1;
• increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- 4 times the number of Red lights;
- 2 times the number of Amber lights;
- 1 times the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- **4 times** the number of Red lights;
- **2 times** the number of Amber lights;
- **1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- **4 times** the number of Red lights;
- **2 times** the number of Amber lights;
- **1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- **4 times** the number of *Red* lights;
- **2 times** the number of *Amber* lights;
- **1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

4 times the number of Red lights;
2 times the number of Amber lights;
1 times the number of current delays

This is derived from the GreenToAmber event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the AmberToRed event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

4 times the number of Red lights;
2 times the number of Amber lights;
1 times the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

4 times the number of Red lights;
2 times the number of Amber lights;
1 times the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
The Variant

The variant is a sum of the following:

- **4 times** the number of Red lights;
- **2 times** the number of Amber lights;
- **1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.
Part III

A general multi-way intersection parametric controller
A General Multi-Way Intersection Parametric Controller
Extending the model

In this development a parameterised version of the preceding traffic light control system is presented. The difference will start to be apparent when you look at the new context in which you will notice that the constant $adir$ is missing.

When you look at the new ChangeLight machine you will notice a few differences. First, there are some new events, or rather one renamed event $ToGreen$, which renames $ChangeLight$, and a new event $ToRed$ that changes a green light to red. Notice that both of these events have parameters, so the new $ToGreen$ and $ToRed$ events are now genuinely parameterised and don’t depend on a single constant value.
In this development a parameterised version of the preceding traffic light control system is presented. The difference will start to be apparent when you look at the new context in which you will notice that the constant $adir$ is missing.

When you look at the new ChangeLight machine you will notice a few differences. First, there are some new events, or rather one renamed event **ToGreen**, which renames **ChangeLight**, and a new event **ToRed** that changes a green light to red. Notice that both of these events have parameters, so the new **ToGreen** and **ToRed** events are now genuinely parameterised and don’t depend on a single constant value.
The change to parameterisation has virtually no effect on the **ChangeLight** machine but has significant affects on the refinement. The refinement now has a variable, \( rdir \), which is used to the parameter for both the ToGreen and ToRed events. Also notice that there are two new variables \( togreen \) and \( tored \). The invariant reveals that both of these variables are of type **BOOL**. The purpose of these two variables is will be explained a little later.

Both **BOOL** variables are initialised to FALSE. Notice that different names have been chosen for the events in the refinement to reduce confusion now that we are refining two abstract operations.

Again, notice that the refinement of the ToGreen event fires only when the conflicting directions have been set to Red, but also notice that this event will only fire if the **BOOL** variable \( togreen \) is TRUE.
The change to parameterisation has virtually no effect on the **ChangeLight** machine but has significant affects on the refinement. The refinement now has a variable, \( rdir \), which is used to the parameter for both the ToGreen and ToRed events. Also notice that there are two new variables \( togreen \) and \( tored \). The invariant reveals that both of these variables are of type \( BOOL \). The purpose of these two variables is will be explained a little later.

Both \( BOOL \) variables are initialised to FALSE. Notice that different names have been chosen for the events in the refinement to reduce confusion now that we are refining two abstract operations.

Again, notice that the refinement of the ToGreen event fires only when the conflicting directions have been set to Red, but also notice that this event will only fire if the \( BOOL \) variable togreen is TRUE.
The change to parameterisation has virtually no effect on the *ChangeLight* machine but has significant affects on the refinement. The refinement now has a variable, \( rdir \), which is used to the parameter for both the ToGreen and ToRed events. Also notice that there are two new variables \( togreen \) and \( tored \). The invariant reveals that both of these variables are of type *BOOL*. The purpose of these two variables is will be explained a little later.

Both *BOOL* variables are initialised to FALSE. Notice that different names have been chosen for the events in the refinement to reduce confusion now that we are refining two abstract operations.

Again, notice that the refinement of the ToGreen event fires only when the conflicting directions have been set to Red, but also notice that this event will only fire if the *BOOL* variable togreen is TRUE.
A new event ToGreenInit appears. The purpose of this event is to initialise the state for the sequence of events that effectively refine the ToGreen event. ToGreenInit can only fire when the BOOL variables togreen and tored are both FALSE.

New events GreenToAmber and AmberToRead produce transitions of the Green conflicting lights through to Red. Each of these events can only fire when togreen is TRUE. Notice a similar initialisation event ToRedInit for the refinement of the ToRed event.

The BOOL variables are required because we are refining two atomic events ToGreen and ToRed and the refinement involves a sequence of events. We need to prevent interference and preserve the atomicity of the ToGreen and ToRed events. The BOOL variables ensure that the complete sequence of events making up a refinement are completed before any other event can be triggered. This problem did not arise in the previous development because the ToGreen event had a fixed parameter and there was no possibility for interference.
A new event **ToGreenInit** appears. The purpose of this event is to initialise the state for the sequence of events that effectively refine the ToGreen event. **ToGreenInit** can only fire when the BOOL variables *togreen* and *tored* are both FALSE.

New events **GreenToAmber** and **AmberToRead** produce transitions of the *Green* conflicting lights through to *Red*. Each of these events can only fire when *togreen* is TRUE. Notice a similar initialisation event **ToRedInit** for the refinement of the **ToRed** event.

The BOOL variables are required because we are refining two atomic events **ToGreen** and **ToRed** and the refinement involves a sequence of events. We need to prevent interference and preserve the atomicity of the **ToGreen** and **ToRed** events. The BOOL variables ensure that the complete sequence of events making up a refinement are completed before any other event can be triggered. This problem did not arise in the previous development because the **ToGreen** event had a fixed parameter and there was no possibility for interference.
A new event \texttt{ToGreenInit} appears. The purpose of this event is to initialise the state for the sequence of events that effectively refine the \texttt{ToGreen} event. \texttt{ToGreenInit} can only fire when the BOOL variables \textit{togreen} and \textit{tored} are both FALSE.

New events \texttt{GreenToAmber} and \texttt{AmberToRed} produce transitions of the \textit{Green} conflicting lights through to \textit{Red}. Each of these events can only fire when \textit{togreen} is TRUE. Notice a similar initialisation event \texttt{ToRedInit} for the refinement of the \texttt{ToRed} event.

The BOOL variables are required because we are refining two atomic events \texttt{ToGreen} and \texttt{ToRed} and the refinement involves a sequence of events. We need to prevent interference and preserve the atomicity of the \texttt{ToGreen} and \texttt{ToRed} events. The BOOL variables ensure that the complete sequence of events making up a refinement are completed before any other event can be triggered. This problem did not arise in the previous development because the \texttt{ToGreen} event had a fixed parameter and there was no possibility for interference.
TrafficLights2_ctx context I

CONTEXT TrafficLights2_ctx

SETS LIGHTS DIRECTION
CONSTANTS

Red
Green
Amber
CONFLICT
TrafficLights2_ctx context III

AXIOMS

\textbf{axm1}: \textit{partition}(LIGHTS, \{Red\}, \{Green\}, \{Amber\})

\textbf{axm2}: \textit{finite}(DIRECTION) \quad \text{DIRECTION is a finite set of directions}

\textbf{axm3}: \textit{CONFLICT} \in DIRECTION \leftrightarrow DIRECTION \quad \text{CONFLICT relates conflicting directions}

\textbf{axm4}: \textit{CONFLICT} \cap (DIRECTION \triangleleft \text{id}) = \emptyset \quad \text{a direction cannot conflict with itself}

\textbf{axm5}: \textit{CONFLICT}^{-1} = \textit{CONFLICT} \quad \text{conflicts are symmetric}

\textbf{thm1}: \quad \forall d \cdot d \in DIRECTION \Rightarrow d \notin \textit{CONFLICT}[\{d\}]

\textbf{thm2}: \quad \forall d1, d2 \cdot d1 \notin \textit{CONFLICT}[\{d2\}] \Rightarrow d2 \notin \textit{CONFLICT}[\{d1\}]

END
MACHINE ChangeLight2
SEES TrafficLights2_ctx
VARIABLES
  \( \text{lights} \)
INVARIANTS
  \( \text{inv1: } \text{lights} \in DIRECTION \to \{ \text{Red, Green} \} \)
  \( \text{inv2: } \forall d \cdot d \in DIRECTION \land \text{lights}(d) = \text{Green} \Rightarrow \text{lights}[\text{CONFLICT} \{d\}] \subseteq \{ \text{Red} \} \)
  \( \text{thm1: } \text{finite}(\text{lights}) \)
EVENTS

Initialisation

begin

act1: \[ \text{lights} : |\text{lights}' \in \text{DIRECTION} \rightarrow \{\text{Red}, \text{Green}\} \]
\[ \land (\forall d \cdot d \in \text{DIRECTION} \land \text{lights}'(d) = \text{Green} \Rightarrow \text{lights}'[\text{CONFLICT}[^{\{d\}}]] \subseteq \{\text{Red}\}) \]

end
ChangeLight2 machine III

Event \( \text{ToGreen} \triangleq \)

any \( adir \)

when

\( \text{grd1: } \) \( \text{lights}(adir) = \text{Red} \)

then

\( \text{act1: } \) \( \text{lights} := \text{lights} \)

\( \triangleq (\text{CONFLICT}[\{ adir \}] \times \{ \text{Red} \}) \)

\( \triangleq \{ adir \mapsto \text{Green} \} \)

end
Event  $\text{ToRed} \equiv$

any  $\text{adir}$

when

$\text{grd1: } \text{lights}(\text{adir}) = \text{Green}$

then

$\text{act1: } \text{lights}(\text{adir}) := \text{Red}$

end

END
MACHINE ChangeLight2R
REFINES ChangeLight2
SEES TrafficLights2_ctx

VARIABLES

xlights
Extended lights, Red, Green and Amber lights
delay
delay between Amber and Red, Red and Green
rdir
current argument of ToGreen or ToRed
togreen
tored
A General Multi-Way Intersection Parametric Controller

ChangeLight2R refinement II

INVARINANTS

inv1: \( xlights \in \text{DIRECTION} \rightarrow \text{LIGHTS} \)

inv2:
\[
\forall d \cdot d \in \text{DIRECTION} \land xlights[{\{d\}}] \subseteq \{\text{Green, Amber}\} \\
\Rightarrow \\
xlights[\text{CONFLICT}[\{d\}]] \subseteq \{\text{Red}\}
\]

inv3: \( \text{togreen} = \text{FALSE} \land \text{tored} = \text{FALSE} \) \\
\Rightarrow \\
\text{lights} = xlights

inv4: \( rdir \in \text{DIRECTION} \)

inv5: \( \text{delay} \subseteq \text{DIRECTION} \)

inv6: \( \text{togreen} \in \text{BOOL} \)

inv7: \( \text{tored} \in \text{BOOL} \)
ChangeLight2R refinement III

inv8: \[ \text{togreen} = \text{TRUE} \Rightarrow \text{tored} = \text{FALSE} \]

inv9: \[ \text{togreen} = \text{TRUE} \Rightarrow \text{lights}(\text{rdir}) = \text{Red} \]

inv10:

\[
\begin{align*}
\text{togreen} = \text{TRUE} \Rightarrow \\
\text{xlights} \Leftrightarrow (\text{CONFLICT}[\{\text{rdir}\}] \times \{\text{Red}\}) \Leftrightarrow \{\text{rdir} \mapsto \text{Green}\} = \\
\text{lights} \Leftrightarrow (\text{CONFLICT}[\{\text{rdir}\}] \times \{\text{Red}\}) \Leftrightarrow \{\text{rdir} \mapsto \text{Green}\}
\end{align*}
\]

thm1: \[ \text{togreen} = \text{TRUE} \]
\[
\Rightarrow \\
\text{CONFLICT}[\{\text{rdir}\}] \Leftrightarrow (\{\text{rdir}\} \Leftrightarrow \text{xlights}) = \\
\text{CONFLICT}[\{\text{rdir}\}] \Leftrightarrow (\{\text{rdir}\} \Leftrightarrow \text{lights})\]
ChangeLight2R refinement IV

\[\text{inv11: } \text{tored} = \text{TRUE} \Rightarrow \text{lights}(\text{rdir}) = \text{Green}\]

\[\text{inv12: } \text{tored} = \text{TRUE} \Rightarrow \text{xlights} \preceq \{\text{rdir} \mapsto \text{Red}\} = \text{lights} \preceq \{\text{rdir} \mapsto \text{Red}\}\]

\[\text{thm2: } \text{tored} = \text{TRUE} \Rightarrow \{\text{rdir}\} \preceq \text{lights} = \{\text{rdir}\} \preceq \text{xlights}\]
ChangeLight2R refinement V

\[ \text{thm3: } \text{finite}(x\text{lights}) \]

\[ \text{thm4: } \forall d, b, a \cdot a \in \text{LIGHTS} \land a \neq b \]

\[ \land x\text{lights}(d) = b \]

\[ \Rightarrow \text{card}((x\text{lights} \leftarrow \{d \mapsto a\}) \triangleright \{b\}) \]

\[ = \text{card}(x\text{lights} \triangleright \{b\}) - 1 \]

changing light in direction \(d\) from \(b\) (= before) to \(a\) (= after) decreases number of colour \(b\) lights by 1

\[ \text{thm5: } \forall d, b, a \cdot a \in \text{LIGHTS} \land a \neq b \]

\[ \land x\text{lights}(d) = b \Rightarrow \]

\[ \text{card}((x\text{lights} \leftarrow \{d \mapsto a\}) \triangleright \{a\}) \]

\[ = \text{card}(x\text{lights} \triangleright \{a\}) + 1 \]

dircthange light in direction \(d\) from \(b\) (= before) to \(a\) (=after) increases number of colour \(a\) lights by 1
ChangeLight2R refinement VI

\textit{thm6}: \forall d, b, a, c \cdot a \in LIGHTS \land c \in LIGHTS \land c \neq a \land c \neq b \land xlights(d) = b \\
\Rightarrow \\
\text{card}((xlights \leftrightarrow \{d \mapsto a\}) \triangleright \{c\}) \\
= \\
\text{card}(xlights \triangleright \{c\}) \\
changing \text{light in direction } d \text{ from } b (\text{= before}) \text{ to } a (\text{=after}) \text{ does not change number of colour } c, c \neq a, c \neq b
EVENTS
Initialisation
begin
with

\text{lights}': \text{lights}' = x\text{lights}'

\text{act1}: x\text{lights} : \{x\text{lights}' \in \text{DIRECTION} \rightarrow \{\text{Red}, \text{Green}\} \wedge (\forall d \cdot x\text{lights}'(d) = \text{Green} \Rightarrow x\text{lights}'[\text{CONFLICT}[^{\{d\}}]] \subseteq \{\text{Red}\})

\text{act2}: \text{delay} := \emptyset

\text{act3}: \text{togreen}, \text{tored} := \text{FALSE}, \text{FALSE}

\text{act5}: r\text{dir} : \in \text{DIRECTION}

end
Event $ToGreen \cong$

refines $ToGreen$

when

grd1: $\text{togreen} = \text{TRUE}$

grd2: $x\text{lights}[\text{CONFLICT}\{\text{rdir}\}] \subseteq \{\text{Red}\}$

grd3: $\text{rdir} \notin \text{delay}$

with

$\text{adir}$: $\text{adir} = \text{rdir}$

then

$\text{act1}$: $x\text{lights}(\text{rdir}) := \text{Green}$

$\text{act2}$: $\text{togreen} := \text{FALSE}$

end
ChangeLight2R refinement IX

Event  $ToGreenInit \triangleq$

  any
  $adir$
when
$grd1$:  $togreen = FALSE$
$grd2$:  $tored = FALSE$
$grd3$:  $xlights(adir) = Red$
then
$act1$:  $rdir := adir$
$act2$:  $togreen := TRUE$
end
ChangeLight2R refinement X

Event  GreenToAmber \Rightarrow
Status  convergent
        any
        dir
when
grd1:  togreen = \text{TRUE}
grd2:  dir \in \text{CONFLICT}[\{rdir\}]
grd3:  xlights(dir) = \text{Green}
grd4:  dir \notin \text{delay}
then
act1:  xlights(dir) := \text{Amber}
act2:  delay := delay \cup \{dir\}
end
ChangeLight2R refinement XI

Event $AmberToRed \equiv$

Status convergent

any

$\text{dir}$

when

$grd1$: $\text{togreen} = TRUE$

$grd2$: $\text{dir} \in CONFLICT[\{rdir\}]$

$grd3$: $\text{xlights}(\text{dir}) = Amber$

$grd4$: $\text{dir} \notin \text{delay}$

then

$act1$: $\text{xlights}(\text{dir}) := Red$

$act2$: $\text{delay} := \text{delay} \cup \{rdir\}$

end
ChangeLight2R refinement XII

Event  \( \text{Delay} \supseteq \)

Status  convergent

  any

  \( \text{dir} \)

when

  \( \text{grd1: } \text{dir} \in \text{delay} \)

then

  \( \text{act1: } \text{delay} := \text{delay} \setminus \{\text{dir}\} \)

end
Event \( ToRed \supseteq \)
refines \( ToRed \)

when

\( grd1: \) \( tored = TRUE \)
\( grd2: \) \( xlights(rdir) = Amber \)
\( grd3: \) \( rdir \notin delay \)

with

\( adir: \) \( adir = rdir \)

then

\( act1: \) \( xlights(rdir) := Red \)
\( act2: \) \( tored := FALSE \)

end
Event \( ToRedInit \triangleq \)

any

\( adir \)

when

\( grd1: \) \( xlights(adir) = \text{Green} \)

\( grd2: \) \( tored = \text{FALSE} \)

\( grd3: \) \( togreen = \text{FALSE} \)

then

\( act1: \) \( rdir := adir \)

\( act2: \) \( tored := \text{TRUE} \)

end
Event $\text{ToAmber} \bowtie$

when

$\text{grd1: tored} = \text{TRUE}$
$\text{grd2:} \quad \text{xlights} (\text{rdir}) = \text{Green}$
$\text{grd3:} \quad \text{rdir} \notin \text{delay}$

then

$\text{act1:} \quad \text{xlights} (\text{rdir}) := \text{Amber}$
$\text{act2:} \quad \text{delay} := \text{delay} \cup \{\text{rdir}\}$

end
A General Multi-Way Intersection Parametric Controller

ChangeLight2R refinement XVI

VARIANT

\[ 4 \times \text{card}(\text{xlights} \rightharpoonup \{\text{Green}\}) + 2 \times \text{card}(\text{xlights} \rightharpoonup \{\text{Amber}\}) + \text{card}(\text{delay}) \]

END